Yazılım Geliştirme Süreci Uyarlama İçin Bir Çerçeve Önerisi

Görkem Giray

İzmir Üniversitesi, Yazılım Mühendisliği (Yarı zamanlı öğretim görevlisi)

gorkem.giray@izmir.edu.tr

Özet. Yazılım geliştirme süreçlerinin gereksinimler doğrultusunda uyarlanmasında rehberlik edecek kapsamlı bir çerçeve bulunmamaktadır. Bu eksiklik birçok araştırmacı tarafından irdelenmiş ve yazılım geliştirme sürecinin uyarlanması için değerlendirilmesi gereken etkenler hakkında çeşitli öneriler verilmiştir. Bu öneriler, bu alandaki bilgi birikimini ortak bir terminoloji oluşturacak şekilde sistemleştirmekten ve bu etkenlerin yazılım geliştirme sürecinde gerçekleştirilen mevcut etkinliklerle bağlantılarını kurmaktan uzaktır. Bu bildiride, (1) yazılım geliştirme sürecini uyarlamak için değerlendirilmesi gereken etkenlerin Essence çerçevesinin çekirdeği genişletilerek ifade edilmesi ve bu etkenlerin nasıl değerlendirileceğine dair önerilerin derlenmesi, (2) yazılım geliştirme sürecinde gerçekleştirilmesi olası etkinliklerin standart biçimde tanımlanması, (3) etkenler ve etkinlikler arasındaki bağlantıların kurularak yazılım geliştirme sürecinin uyarlanmasına yardımcı olacak önerilerin oluşturulması için bir yol haritası önerilmektedir. Bu yolda kullanılacak terminoloji için yazılım geliştirme sürecindeki önemli kavramları modelleyen Essence çerçevesi temel alınmıştır.

Anahtar Kelimeler: Yazılım geliştirme süreci, süreç uyarlama, Essence çerçevesi.

1 Giriş

Modern dünyada yazılımın hayatımızın ayrılmaz bir parçası haline geldiği kabul edilmektedir. Elimizdeki telefonda, evimizdeki televizyonda, kullandığımız arabada, bindiğimiz uçakta, tedavi gördüğümüz hastanede, kısacası neredeyse her yerde, yazılım insanın bazı gereksinimlerini karşılaması için önemli bir araç olarak karşımıza çıkmaktadır. Yazılımın hayatımızdaki bu önemi, çıktısı yazılım olan yazılım geliştirme sürecini de oldukça önemli kılmaktadır.

Bir süreç, girdileri dönüştürerek çıktı üretmektedir [1]. Aynı şekilde bir yazılım geliştirme süreci de gereksinimleri girdi olarak alıp bunları işleyerek, dönüştürerek bir yazılım üretmektedir [2]. Bu üretim sürecinde girdi ve süreç içindeki bileşenlerin davranışı ne kadar tahmin edilebilir olursa çıktı da o kadar öngörülebilir olmaktadır. Süreç içinde öngörülebilirliği arttıran en önemli etken olarak otomasyonun çok ve insanın karar almasını gerektirecek durumların ise az olması sayılmaktadır. Örneğin,

önceden tanımlanmış girdilerin, daha çok üretim makinelerinin ve iyi tanımlanmış etkinlikleri uygulayan insanların bulunduğu bir sürecin ürettiği çıktıların nitelikleri ve niceliği yüksek seviyede tahmin edilebilir olmaktadır. Yazılım geliştirme sürecinde de otomasyonun görece az ve insanın karar aldığı durumların görece fazla olduğu görülmektedir. Bununla paralel olarak yazılım geliştirme sürecinin çıktısı olan yazılımın özünde bulunan karmaşıklık, uyumluluk, değişebilirlik ve görünmezlik özellikleri de [3] süreçte yaşanabilecek birçok zorluğa işaret etmektedir.

Yazılımın özünde bulunan özelliklerin getirdiği zorluğun yanında (1) yazılım mühendisliğinde kapsayıcı teorilerin eksikliği [4, 5, 6], (2) her yazılım geliştirme projesinin farklı dinamiklerinin olması ve (3) yazılım geliştirme sürecindeki etkinliklerin çoğunun endüstri tarafından geliştirilmesi [7] yazılım geliştirme sürecine daha sistemli bir yaklaşıma gereksinim olduğuna işaret etmektedir.

Yazılım mühendisliğinde kapsayıcı teorilerin eksikliği ve buna duyulan gereksinim bazı çalışmalarda dile getirilmiştir [4, 5, 6]. Benzer şekilde yazılım geliştirme süreci için de kapsayıcı, bilgi ve deneyim birikimini birleştirici bir teoriye ihtiyaç duyulmaktadır [8]. Temelde bu teori yokluğu nedeniyle yazılım geliştirme sürecinde kullanılan etkinliklerin önemli bir bölümünün tasarımları geçmişteki deneyimlere dayanmaktadır [9]. Bu deneyimler, belirli koşullar altında sağduyu ile yapılan tercihler ile şekillenmektedir [10]. Proje yönetimi çalışma alanında her projenin biricik olduğu ve takım, paydaşlar gibi birçok boyuta göre değişkenlik gösterdiği [11, 12], yazılım mühendisliği alanında da benzer şekilde her projenin farklı dinamikleri olduğu [13, 14, 15, 16] genel olarak kabul edilmektedir. Bu kabul, yapılan tercihleri etkileyen etkenlerin oluşturduğu durumların çokluğunun bir göstergesidir. Bu kadar çeşitliliğin olduğu bir ortamda yazılım geliştirme sürecinde gerçekleştirilen etkinliklerin genellikle endüstride tasarlanması da [7] ortak bir kavramlar bütününün ortaya çıkmasını zorlaştırmaktadır. Bunun temel nedeninin endüstrideki en önde gelen amacın kar etmek olduğu ve bilimin ilerlemesine katkıda bulunmanın kar etme amacının yanında çok daha düşük bir öneme sahip olduğu söylenebilir. Dolayısıyla elde edilen deneyimlerin nesnel biçimde değerlendirilerek genel olarak kabul görmüş bir terminolojiyle paylaşılması endüstrideki ana amaçlardan biri değildir. Bu da endüstride elde edilen deneyimlerin paylaşılmasını ve karşılaştırılmasını çok zorlaştırmakta hatta bazı durumlarda imkansız kılmaktadır.

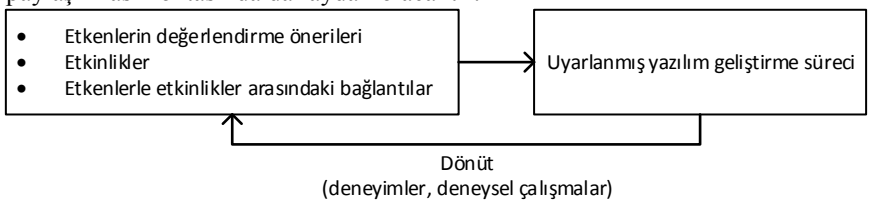
Cockburn, yazılım mühendisliği alanında deneyimlerin harmanlanmasından elde edilen modellerin aşağıdaki dört önemli gereksinimi karşılama konusunda eksik kaldığını bildirmektedir [17]:

1. Yazılım geliştirme projeleri için önemli olan etkenlerin ve bunlar arasındaki ilişkilerin belirlenmesi,
2. Yazılım geliştirme projelerinde elde edilen sonuçların nedenlerinin sistematik olarak anlaşılması,
3. Ortak bir terminolojinin oluşması,
4. Yazılım geliştirme projelerinde gerçekleştirilecek etkinliklerin belirlenmesi konusunda geçmiş deneyimlerin değerlendirilmesi.

Endüstrideki deneyimlerin, kar ya da başka bir öznel fayda elde etme amacı gütmeden, nesnel olarak değerlendirilebilmesi, paylaşılabilmesi ve karşılaştırılabilmesi önemli bir gereksinimdir. Bu gereksinimin karşılanabilmesi için ilk adım elde edilen

deneyimlerin ortak tanımlar, kavramlar etrafında sistemleştirilmesidir; yani ortak, kapsamlı bir terminolojinin oluşturulmasıdır. Bu sistemli yaklaşım, gerek endüstride gerekse akademide üretilen bilginin nesnel biçimde değerlendirilmesini, verimli biçimde paylaşılmasını ve öğretilmesini, evrilmesini sağlayacaktır; kısacası bu bilgi birikiminin değerini arttıracaktır.

Bu bildiride bir yazılım geliştirme sürecinde gerçekleştirilecek etkinlikler seçilirken değerlendirilmesi önerilen etkenler, yazılım geliştirme işinin özündeki temel kavramları ve bu kavramlar arasındaki ilişkileri modelleyen Essence çerçevesinin çekirdeği [18] genişletilerek sunulmuştur. Buradaki temel amaç ortak ve kapsamlı bir terminoloji geliştirmektir. Bu terminoloji, yazılım geliştirme sürecinde kullanılan etkinliklerin (eşli programlama [19], Sprint planlama [20], Günlük Scrum [20], Sprint değerlendirme [20], iterasyon planlama [21], kullanım senaryolarının ve aktörlerin bulunması [21] gibi) standartlaştırılmasıyla ve etkenlerin değerlendirme sonuçlarıyla etkinlikler arasındaki bağlantıların kurulmasıyla (örneğin, küçük takımlarda günlük toplantıların yapılması gibi) Şekil 1'de gösterildiği gibi yazılım geliştirme sürecinin uyarlanması için yol gösterici olacaktır. Bu terminoloji üzerine bina edilmiş olan bilgi birikimi de deneyimler ve deneysel çalışmalar doğrultusunda iyileşecek ve genişleyecektir. Bu ortak terminoloji daha fazla deneyimin belgelenmesi ve paylaşılması noktasında da faydalı olacaktır.



Şekil 1. Yazılım geliştirme sürecinin uyarlanması

Bildirinin ikinci bölümünde literatürdeki ilgili çalışmalardan örnekler verilmiş, üçüncü bölümünde Essence çerçevesi tanıtılmış, dördüncü bölümünde Essence çerçevesi genişletilerek bir yazılım geliştirme sürecinin uyarlanması için nasıl bir çerçeve önerilebileceği konusunda bir yol haritası sunulmuştur. Böyle bir çerçevenin oluşturulması durumunda bu çerçevenin nasıl kullanılabileceğine bir örnek yine dördüncü bölümde verilmiş, beşinci bölümde sonuçlar ve gelecekte yapılması planlanan çalışmalar sunulmuştur.

2 İlgili Çalışmalar

Boehm ve Turner, bir yazılım geliştirme yöntemi seçmek için beş etkenin değerlendirilmesi gerektiğini belirtmişlerdir [22]. Bu beş etken değerlendirilerek çevik yöntemlerin önerdiği etkinliklerin disipline edilmesi, kısacası mevcut koşullar doğrultusunda yazılım geliştirme sürecinin özelleştirilmesi önerilmektedir.

Çevik yazılım geliştirme yöntemleri şemsiyesi altında sınıflandırılan Crystal yöntem ailesinde gerçekleştirilecek etkinliklerin seçimi için kullanılan iki temel boyut

takım büyüklüğü ve yazılım hatalarına karşı tolerans olarak belirlenmiştir [23]. Takım büyüklüğü, yazılım geliştirme için önemli olan iletişim ve koordinasyon gereksinimleri için önemli bir gösterge olarak ele alınmaktadır. Yazılım hatalarına karşı tolerans ise yazılım sistemindeki bir hatadan dolayı kaynaklanabilecek bir rahatlığın kaybı, maddi kayıplar ve insan hayatının olumsuz etkilenmesi gibi bir yelpazeyi temsil etmektedir. Bu iki boyuta göre Crystal yöntem ailesindeki Crystal Clear, Crystal Yellow, Crystal Orange ve Crystal Red yöntemlerinden birisinin seçimi yapılmaktadır.

Ambler çevik yazılım geliştirme yöntemlerinin büyük ölçekli projelerde kullanılmasında değerlendirilmesi gereken dokuz etken önermiştir [24]. Ambler'in çalışmasının devamı niteliğindeki bir başka çalışmada, Ambler ve Lines çevik yazılım geliştirme yöntemlerinin önerdiği etkinlikler kullanılarak mevcut koşullara uygun uyarlanmış bir çevik yazılım geliştirme yönteminin elde edilebileceğini belirtmişlerdir [25].

Kruchten, çevik yazılım geliştirme etkinliklerinin seçimi için değerlendirilmesi gereken etkenleri içeren bir bağlamsal model önermiştir [26]. Bu modelde, organizasyonel seviyedeki beş etken proje seviyesindeki sekiz etkeni etkilerken bu 8 etken de projede gerçekleştirilecek etkinlikleri belirlemede yol gösterici olacağı belirtilmektedir.

Clarke ve O'Connor, en uygun yazılım geliştirme sürecinin, yazılım geliştirme ortamının durumsal özelliklerine bağlı olduğunu belirtmişlerdir [27]. Bu durumsal özellikleri literatürdeki çalışmalardan derleyerek 44 etken olarak listelemişler ve bu etkenleri 8 başlık altında sınıflandırmışlardır. Bu etkenlerin nasıl değerlendirileceği ve bu değerlendirme sonucunda neler yapılabileceği konularında herhangi bir yönlendirme bulunmamaktadır.

Klaus ve Kuhrmann, yazılım geliştirme sürecinin uyarlanması için değerlendirilmesi gereken proje özelliklerini 49 etkenden oluşan bir katalogda toplamışlardır [14]. Bu çalışmada etkenlerin değerlendirilmesi sonucunda atılabilecek adımlar çok genel öneriler (örneğin kullanıcı katılımını yoğunlaştır, sistem mimarisi üzerine çalışmaları yoğunlaştır, belgelemeyi arttır, belgelemeyi azalt, iterasyon sayısını arttır gibi) şeklinde verilmiştir.

Tablo 1'de ilgili çalışmalarda belirtilen etkenlerin bazıları listelenmiştir.

Yazılım geliştirme yöntemlerinin özelleştirilmesine yönelik çalışmaların yanında bir yazılımın yaşam döngüsü için standart bir terminoloji oluşturma amacı güden standartlar da bulunmaktadır. IEEE STD 12207-2008 standardı bir yazılımın tüm yaşam döngüsünü tanımlamak için genel bir süreç çerçevesi önermektedir [28]. Bu standardın amacı yazılım sisteminin paydaşları arasındaki iletişimi kolaylaştırmak için standart süreçler, yani ortak bir terminoloji, sağlamaktır [28]. Bu standartta belirtilen süreçlerin risk, karmaşıklık, büyüklük gibi çeşitli etkenlere göre uyarlanması gerektiği belirtilmiştir [1].

3 Essence Çerçevesi

Essence çerçevesi, SEMAT (Software Engineering Method and Theory) girişimi tarafından yazılım geliştirmek için kullanılan yöntemlerin standartlaştırılması

amacıyla oluşturulmuştur. Şekil 2'de gösterildiği gibi çerçeve 4 temel bileşenden oluşmaktadır [7]:

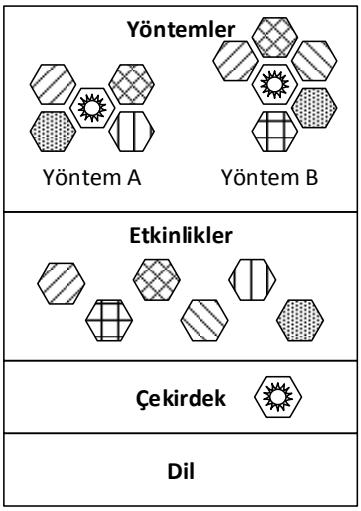
Tablo 1. Yazılım geliştirme süreci uyarlama için değerlendirilmesi önerilen etkenlerin kısmi listesi

Etken	Etkenin belirtildiği çalışmalar	Etkenin açıklaması
Paydaşların yazılım geliştirme deneyimi	[14, 27]	Yazılım geliştirme süreci ve paydaşların bu süreçteki rolü hakkında genel bilgi sahibi olma; beklentilerinin gerçekçiliği hakkında öz değerlendirme yapabilme
Paydaşların ulaşılabilirliği	[14, 27]	Paydaşlara bilgi almak için ulaşabilme kolaylığı
Paydaşların sayısı	[14, 27]	Yazılım geliştirme sürecinin farklı aşamalarında bir role sahip olan kişi sayısı
Fırsatın zamanlaması	[27]	Fırsatın ne kadar acil değerlendirilmesi gerektiği
Gereksinimlerin yapılabilirliği	[27]	Gereksinimlerin karşılanması için gerekli teknik unsurların varlığı
Gereksinimlerin değişim sıklığı	[19, 27]	Gereksinimlerin ne sıklıkla, ne kadar değiştiği
Yazılım sisteminin karmaşıklığı	[24, 27]	Yazılım sisteminin, sistemi oluşturan alt sistemlerin anlaşılabilme kolaylığı
Yazılım sistemi – Yeni/Yenileme	[14, 24, 26]	Geliştirilecek yazılım sisteminin mevcut bir sistemin yerine geçip geçmeyeceği
Takımın büyüklüğü	[14, 19, 23, 24, 26, 27]	Takımda yer alan kişi sayısı
Takımın kültürü	[19, 24, 26, 27]	Takımın iş yapma biçimini etkileyen çeşitli unsurlardan meydana gelebilen birikim; özellikle değişime, düzene uyum sağlama yeteneği
Takımın alan bilgisi	[14, 19, 23, 27]	Takımın yazılım sisteminin kullanılacağı alan hakkındaki bilgi seviyesi
Takımın teknik bilgisi	[14, 19, 23, 27]	Takımın yazılım geliştirme işi hakkındaki bilgi seviyesi
Yazılım hatalarına karşı tolerans	[19, 23, 26, 27]	Yazılım sisteminde çıkabilecek hatalarda nasıl zararların oluşabileceği
Gereksinimlerin karşılanmasındaki esneklik	[27]	Gereksinimlerin nasıl karşılanacağı konusunda ne kadar esneklik olabileceği
Yönetim desteği	[27]	Yönetimin, üst düzey yöneticilerin projeye destek seviyesi
Paydaşların takımı destekleme tarzı	[27]	Paydaşların takımı destekleme tarzı: otoriter, zorlayıcı, samimi, vs.

- *Dil:* Çekirdeği, etkinlikleri ve yöntemleri tanımlamak için kullanılan dil ögelerini ve bu ögeler arasındaki ilişkileri içerir [18]. Dil ögelerine örnek olarak alfa, alfa

durumu, iş ürünü, etkinlik uzayı, etkinlik ve yetkinlik; bu ögeler arasındaki bir ilişkiye de bir etkinliğin bazı yetkinlikleri gerektirmesi verilebilir [18].

- *Çekirdek:* Yazılım geliştirme sürecinde bulunan, etkinliklerden ve yöntemlerden bağımsız olarak bu süreçteki önemli kavramları içerir [18]. Bu kavramları ifade etmek için alfa kavramı kullanılır [18]. Alfalar, (1) yazılım geliştirme sürecinin temelinde bulunan ortak kavramları temsil eder; (2) yazılım geliştirme sürecinin nasıl ve ne kadar sağlıklı ilerlediğinin izlenmesini ve değerlendirilmesini sağlar; (3) yazılım mühendisliğindeki yöntemleri ve etkinlikleri tanımlamak için bir kavramlar temeli sağlar [18].
- *Etkinlikler:* Bir etkinlik yazılım geliştirme sürecinin bir yönünün nasıl ele alınacağını tanımlar. Bir etkinlik, net bir amaç ve bu amaca ulaşabilmek için çeşitli yönlendirmeler içermektedir. Etkinlik için bir örnek olarak gereksinimlerin belirtimi için kullanıcı hikayelerinin kullanımı gösterilebilir [29].
- *Yöntemler:* Bir yöntem ise çekirdeğin ve bunun etrafında konumlandırılan etkinliklerden oluşur [18].



Şekil 2. Yöntem mimarisi [7]

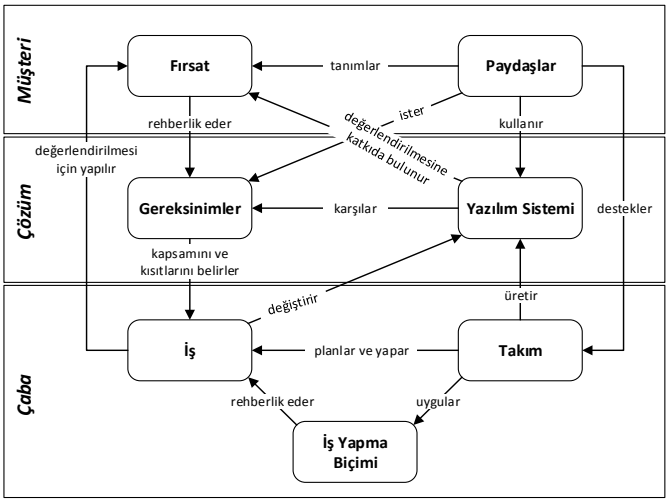
3.1 Çekirdek

Çekirdek, yazılım geliştirme sürecinin temelinde bulunan kavramları (alfalar) ve bu alfalar arasındaki ilişkileri tanımlamaktadır. Bu alfaların ortak bir terminoloji oluşturarak takım içindeki iletişimi daha verimli kılmada, zorlukları çözümlemede ve yazılım geliştirme sürecinin verimini arttırma konusunda yardımcı olabileceği belirtilmektedir.

Çekirdekte bulunan alfalar üç temel ilgi alanına ayrılmıştır [18]: (1) Müşteri ilgi alanı yazılım sisteminin canlı ortamda kullanımıyla, (2) çözüm ilgi alanı yazılım sisteminin belirtimi ve geliştirilmesiyle, (3) çaba ilgi alanı ise takım ve takımın iş yapma biçimiyle ilgili kavramları kapsamaktadır.

Çekirdekte bulunan alfalar ve bu kavramlar arasındaki ilişkiler Şekil 3'te gösterilmiştir [18]. Yedi temel alfa üç ilgi alanında (müşteri, çözüm ve çaba) gösterilmiştir.

1. *Fırsat:* Yazılım sistemini geliştirmek ya da değiştirmek için oluşan durumu, bir başka deyişle paydaşların iş gereksinimlerinin tüm takım tarafından anlaşılmış halini temsil eder. Fırsat, yazılım sisteminin geliştirilmesi için detaylı gereksinimlerin oluşturulmasına rehberlik eder.
2. *Paydaşlar:* Yazılım sistemini etkileyecek ya da yazılım sistemi tarafından etkilenecek kişileri, grupları, organizasyonları temsil eder. Paydaşlar, (1) fırsatı tanımlar; (2) detaylı gereksinimlerin belirlenmesinde temel bilgi kaynağı olarak davranır; (3) yazılım sistemini doğrudan kullanır ya da bu kullanımdan bir şekilde etkilenir; (4) takımı yazılım geliştirme sürecinde destekler.
3. *Gereksinimler:* Yazılım sisteminin yapacaklarını ve bunları hangi koşullar altında ve kalitede yapacağını tanımlar. Gereksinimler yapılacak işlerin kapsamını ve kısıtlarını belirler.
4. *Yazılım sistemi:* Yazılımdan, donanımdan ve veriden oluşan sistemdir. Yazılım sistemi (1) fırsatın değerlendirilmesine katkıda bulunur; (2) gereksinimleri karşılar.
5. *İş:* Bir sonucu elde etmek için yapılan zihinsel ya da fiziksel etkinliktir. İşler, (1) fırsatın tanımladığı duruma uygun olarak belirlenir; (2) yazılım sistemini değiştirir.
6. *Takım:* Bir yazılım sisteminin geliştirilmesinde ve bakımında görev alan kişilerin oluşturduğu gruptur. Bir ya da birden fazla takım (1) yazılım sistemini geliştirir ya da değiştirir; (2) işleri planlar ve yapar; (3) işleri yaparken çeşitli iş yapma biçimlerini kullanır.
7. *İş yapma biçimi:* Takımın gerçekleştirdiği etkinlikler ve kullandığı araçlardır. İş yapma biçimi, yapılan işlerin nasıl yapılacağına rehberlik eder.



Şekil 3. Çekirdekte bulunan kavramlar ve bu kavramlar arasındaki ilişkiler [18]

3.2 Çekirdeğin Genişletilmesi

Ng ve arkadaşlarının önerdiği gibi çekirdekteki alfaların özellikleri belirlenerek çerçevenin genişletilmesi mümkündür [30]. Takım alfasının özelliklerine büyüklük ve deneyim, yazılım sistemi alfasının özelliklerine ise büyüklük ve karmaşıklık örnek olarak verilebilir [30]. Bunun yanında alfalar arasındaki ilişkilerin de özellikleri tanımlanabilmektedir. Takım ve yazılım sistemi alfalarının arasındaki ilişkinin bir özelliği olarak takımın yazılım sistemini sahiplenme seviyesi tanımlanabilir [30].

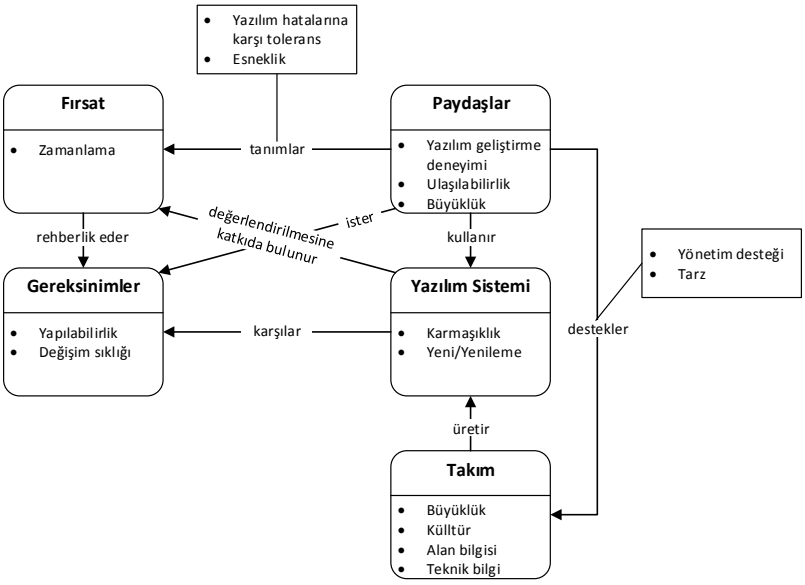
4 Yazılım Geliştirme Süreci Uyarlama için Bir Çerçeve Geliştirilmesi

Bir yazılım geliştirme sürecinin uyarlanmasında rehberlik edecek, Essence çerçevesi kullanılarak bir çerçevenin oluşturulması için aşağıdaki adımların gerçekleştirilmesi önerilmektedir.

- Alfaların ve alfalar arasındaki ilişkilerin özellikleri olarak yazılım geliştirme sürecinin uyarlanmasını etkileyen etkenlerin belirlenmesi
- Etkenlerin nasıl değerlendirileceğine dair yönlendirmelerin derlenmesi
- Etkinliklerin standart bir biçimde tanımlanması
- Etkenler ve etkinlikler arasında bağlantıların kurularak yazılım geliştirme sürecinin uyarlanması için önerilerin oluşturulması
- Ortak terminoloji kullanılarak yapılan deneylerden ve deneyim paylaşımlarından gelen dönütler doğrultusunda etkenlerin, etkinliklerin, etken ve etkinlik arasındaki bağlantıların, uyarlama önerilerinin düzeltilmesi, zenginleştirilmesi

Ng ve arkadaşlarının çalışmasından [30] yola çıkılarak Essence çerçevesinin çekirdeğinin ilgili çalışmalarda önerilen etkenler kullanılarak genişletilmesi ve bu genişletilmiş çerçevenin yazılım geliştirme sürecindeki etkinliklerin seçiminde temel olarak kullanılması hedeflenmektedir. Bu doğrultuda ilgili çalışmalarda bulunan etkenlerin bazılarının çekirdekte bulunan alfalar ya da alfalar arasındaki ilişkilerle, Şekil 4'te gösterildiği gibi, bağlantıları kurulmuştur.

Etkenlerin nasıl değerlendirileceğine dair literatürdeki çalışmalar taranarak bir bilgi tabanı derlenecektir. Bazı etkenler için çeşitli disiplinlerden teoriler, bazıları için uç noktaların belirlendiği yelpazeler, bazıları içinse daha yapısal olmayan deneyimler bilgi tabanını oluşturacaktır. Örneğin, takımların büyüklüğü etkeni için Cockburn 10'a kadar kişiden oluşan takımları küçük, 10 ile 40 arasında kişiden oluşan takımları orta ölçekli olarak nitelendirmektedir [9]. Ambler 10-15 ya da daha az kişiden takımları küçük, 10-15 ile 30-40 arasında sayıda kişiden oluşan takımları orta ölçekli, 30-40 kişiden fazla kişi içeren takımları ise büyük ölçekli olarak nitelendirmektedir [31]. Yazılım sisteminin karmaşıklığı tek platformdaki basit bir sistemden çok platformdaki karmaşık bir sisteme kadar uzanabilir. Yazılım sisteminde oluşan bir problemden dolayı karşılaşılabilecek durum bir rahatlığın kaybedilmesinden, maddi zarara ya da insanın hayatını kaybetmesine kadar gidebilir [9]. Bu duruma göre yazılım sistemindeki hatalara karşı tolerans seviyesi belirlenebilir.



Şekil 4. Essence çerçevesinin çekirdeğinin etkenlerle genişletilmesi (kısmi)

Yazılım geliştirme süreçlerinde gerçekleştirilen çok sayıda etkinlik bulunmaktadır. Örnek olarak eşli programlama [19], sürekli bütünleştirme [19], Sprint planlama [20], Günlük Scrum [20], Sprint değerlendirme [20], Sprint süreç gözden geçirme toplantısı [20], iterasyon planlama [21], kullanım senaryolarının ve aktörlerin bulunması [21], tasarımın gözden geçirilmesi [21] gibi bir süreç modeline özgü ya da bir süreç modelinde yer alan etkinliklerin yanında resmi gözden geçirme, çalıştay, mülakat gibi daha genel etkinlikler de verilebilir. Bu etkinlikler derlenirken her etkinliğin girdisinin, çıktısının, giriş ve çıkış ölçütlerinin, paydaşların rollerinin belirlenmesi gerekmektedir. Bunun yanında etkinliklerin detay seviyeleri birbirinden farklı olabileceği için bu seviyeler dikkate alınmalı ve etkinlikler arasındaki ilişkiler de belirlenmelidir. Etkinliklerin tanımlanması için IEEE STD 12207-2008 [28] gibi standartlardan faydalanılacaktır.

Etkenler, etkenlerin nasıl değerlendirileceği konusundaki yönlendirmeler belirlendikten ve etkinlikler standart olarak tanımlandıktan sonra etkenlerle etkinlikler arasındaki bağlantılar literatürde yapılan çalışmalara dayanılarak belirlenebilir.

4.1 Örnek Bir Durum Çalışması

Tablo 2'de örnek bir yazılım geliştirme projesinde yapılan etken değerlendirme sonuçlarının hangi etkinlik önerilerine dönüşebileceği gösterilmektedir.

Yazılım geliştirme sürecinde gerçekleştirilecek etkinliklerin belirlenmesinde bir çerçeveye dayanarak yapısal bir analizin yapılması sürecin gereksinimlere uygun biçimde uyarlanmasını, bazı etkenlerin değerlendirilmeme riskini ortadan kaldırılmasını, deneyimlerden faydalanılmasını ve elde edilen deneyimlerin anlaşılabilir, kıyaslanabilir biçimde paylaşılabilmesini sağlayacaktır.

Tablo 2. Etkenlerin değerlendirilmesi ve olası etkinliklerin belirlenmesi

Etken değerlendirmesi	**Etkinlik önerisi**
Paydaşlar, yazılım geliştirme deneyimi: Paydaşlar, gerçekçi beklentiler oluşturacak ve üzerlerine düşecek görevleri gereğince getirecek kadar yazılım geliştirme süreci hakkında bilgi sahibi değil	Önemli paydaşlara yazılım geliştirme süreci hakkında bilgi verilmesi [32]
Paydaşlar, büyüklük: Süreçte karar alabilecek, katkı yapabilecek paydaş sayısı 40 civarında *Paydaşlar, ulaşılabilirlik:* Genelde önceden planlamak koşuluyla kısıtlı zamanlar için ulaşılabilir	Önemli kararların ve bilgilerin belgelenmesi ve yazılı olarak paylaşılması [33] Önemli paydaşlardan bilgi almak ve vermek için çalıştayların düzenlenmesi [34]
Paydaşlar, büyüklük: Gereksinimleri belirlemek, çözümlemek ve geçerlemek için çok sayıda kullanıcı ile görüşülmesi gerekiyor	Kullanıcı temsilcilerinden gereksinimlerin toplanması için çalıştayların düzenlenmesi [34]
Yazılım Sistemi, yeni/yenileme: Mevcut çalışan bir sistemin yerine tamamen yeni bir sistem gelecek	Mevcut sistemin paydaşlar tarafından olumlu ve olumsuz bulunan işlevleri ve özellikleri analiz edilmeli [35]
Yazılım Sistemi, karmaşıklık: İşlev sayısı fazlalığından gelen bir karmaşıklık var, teknik bir karmaşıklık ya da belirsizlik yok *Takım, alan bilgisi:* Yeterli seviyede *Takım, teknik bilgi:* Yeterli seviyede	Yazılım sistemi için prototip geliştirmeye gerek yok; kağıt üzerinde basit prototip çizimlerle paydaşlardan geri bildirim alınabilir [34]
Paydaş – Takım, tarz: Otoriter	Resmi, düzenli ve sıklıkla yazılı bilgilendirme yapılması
Gereksinim – Yazılım Sistemi, yazılım hatalarına karşı tolerans: Temel operasyonları etkileyen süreçleri etkilediği için yüksek risk	Planlı olarak belirli aralıklarla (4-6 haftada bir) kullanıcı testlerinin yapılması

5 Sonuç

Bu bildiri, bir yazılım geliştirme sürecinin bazı etkenler değerlendirilerek uyarlanmasında kullanılabilecek bir çerçevenin oluşturulma gerekliliğini ortaya koymaktadır. Bu çerçevenin en önemli görevi yazılım geliştirme sürecini oluşturan ögelerin ortak bir dille ifade edilmesini sağlamaktır. Bu ortak dilin temel kavramlarının Essence çerçevesinin çekirdeğinde bulunan alfalar tarafından karşılanabileceği düşünülmektedir. Bu doğrultuda literatürdeki bazı çalışmalarda yazılım süreci uyarlama için önerilen etkenler incelenmiş ve bu etkenlerden bazıları alfalar ve alfalar arasındaki ilişkilerin özellikleri olarak tanımlanmıştır. Daha sonra böyle bir çerçevenin olması durumunda yazılım sürecinde gerçekleştirilecek etkinliklerin nasıl seçilebileceği konusunda basit bir senaryo sunulmuştur.

5.1 Kısıtlar

Bu çalışma, literatürde bulunan etkenleri ele almakta ve bu etkenleri Essence çerçevesini kullanarak düzenlemektedir. Yeni bir etken keşfetmek ya da bu etkenlerin etkinliklerle nasıl bir bağlantısının bulunduğu konusunda deney yapmak bu çalışmanın kapsamının dışındadır.

5.2 Gelecek Çalışmalar

Literatürde belirtilen etkenler daha geniş ölçekli bir tarama yapılarak derlenecek ve Essence çerçevesi kullanılarak düzenlenecektir. Benzer şekilde yazılım geliştirme sürecinde gerçekleştirilen etkinliklerin bir listesi hazırlanarak bu etkinliklerin standart birer tanımı yapılacaktır. Daha sonra yine literatürdeki deneysel çalışmalardan faydalanılarak etkenler ile etkinlikler arasındaki bağlantılar kurulacaktır. Sonuç olarak elde edilen çerçevenin kullanımı sırasında elde edilecek dönütler doğrultusunda çerçeve iyileştirilecek ve zenginleştirilecektir.

6 Kaynakça

1. Systems and software engineering – System life cycle processes, ISO/IEC 15288:2008(E) IEEE Std 15288-2008 (Revision of IEEE Std 15288-2004) (2008)
2. Systems and software engineering – Vocabulary, ISO/IEC/IEEE 24765:2010(E) (2010)
3. Brooks, F.P., Jr.: No Silver Bullet Essence and Accidents of Software Engineering. Computer, Vol. 20, No. 4. (1987) 10-19
4. Johnson, P., Ekstedt, M., Jacobson, I.: Where's the Theory for Software Engineering?. Software, IEEE, Vol. 29, No. 5. (2012) 96
5. Hannay, J.E., Sjoberg, D.I.K., Dyba, T.: A Systematic Review of Theory Use in Software Engineering Experiments. IEEE Trans. Softw. Eng., Vol. 33, No. 2. (2007) 87-107
6. Johnson, P., Ekstedt, M.: In Search of a Unified Theory of Software Engineering. International Conference on Software Engineering Advances - ICSEA 2007. (2007) 1
7. Péraire, C.: A Step Forward in Software Engineering Education: Introducing the SEMAT Essence Framework. Keynote Address - LACREST 2013. Medellin. (2013)
8. Ralph, P.: Evaluating process theories in software engineering. In Proc. of the 3rd SEMAT Workshop on General Theories of Software Engineering. ACM, USA (2014) 5-8
9. Cockburn, A.: Agile Software Development: The Cooperative Game, 2nd Ed. Addison Wesley Professional (2006)
10. Maier, M.W., Rechtin, E.: The Art of Systems Architecting, 2nd Ed. CRC Press (2000)
11. Managing Successful Projects with PRINCE2 2009 Edition Manual. Office of Government Commerce, The Stationery Office (2009)
12. A Guide to the Project Management Body of Knowledge: PMBOK Guide, 5th Ed. Project Management Institute (2013)
13. Rong, G., Boehm, B., Kuhrmann, M., Tian, E., Lian, S., Richardson, I.: Towards context-specific software process selection, tailoring, and composition. In Proc. of the 2014 Int. Conf. on Software and System Process (ICSSP 2014). ACM, USA (2014) 183-184
14. Kalus, G., Kuhrmann, M.: Criteria for software process tailoring: a systematic review. In Proceedings of the 2013 International Conference on Software and System Process (ICSSP 2013). ACM, New York, NY, USA (2013) 171-180

15. Jones, C.: Development Practices for Small Software Applications. CrossTalk, The Journal of Defense Software Engineering, Vol. 21, No. 2 (2008) 9-13
16. MacCormack, A., Verganti, R.: Managing the Sources of Uncertainty: Matching Process and Context in Software Development. Journal of Product Innovation Management, Vol. 20 (2003) 217–232
17. Cockburn, A.: The end of software engineering and the start of economic-cooperative gaming. Computer Science and Information Systems, Vol. 1, Issue 1 (2004) 1-32
18. Kernel and Language for Software Engineering Methods (Essence), OMG (2014)
19. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change, 2nd Ed. Addison Wesley Professional (2004)
20. Schwaber, K.: The Enterprise and Scrum. Microsoft Press (2007)
21. Kruchten, P.: The Rational Unified Process: An Introduction, 3rd Ed. Addison Wesley (2003)
22. Boehm, B., Turner, R.: Observations on Balancing Discipline and Agility. Agile Development Conference (2003) 32-39
23. Cockburn, A.: Crystal Clear A Human-Powered Methodology for Small Teams. Addison Wesley Professional (2004)
24. Ambler, S.W.: Agile Software Development at Scale. In: Meyer, B., Nawrocki, J.R., Walter B. (eds.): Balancing Agility and Formalism in Software Engineering. Lecture Notes In Computer Science, Vol. 5082. Springer-Verlag, Berlin, Heidelberg (2008) 1-12
25. Ambler, S.W., Lines, M.: Scaling Agile Software Development: Disciplined Agility at Scale. Disciplined Agile Consortium White Paper Series (2014) http://disciplinedagileconsortium.org/Resources/Documents/ScalingAgileSoftwareDevelopment.pdf (28.04.2015 tarihinde erişildi)
26. Kruchten, P.: Contextualizing agile software development. Journal of Software: Evolution and Process Vol. 25, No. 4 (2013) 351-361
27. Clarke, P., O'Connor, R.V.: The situational factors that affect the software development process: Towards a comprehensive reference framework. Inf. Softw. Technol. Vol. 54, No. 5 (2012) 433-447
28. Draft IEEE International Standard - Systems and software engineering - Software life cycle processes, ISO/IEC/IEEE P12207-CD2-1410 (2015)
29. Elvesæter, B., Benguria, G., Ilieva. C.: A comparison of the Essence 1.0 and SPEM 2.0 specifications for software engineering methods. In Proc. of the Third Workshop on Process-Based Approaches for Model-Driven Engineering (PMDE 2013). ACM, USA (2013)
30. Ng, P., Huang, S., Wu, Y.: On the value of essence to software engineering research: A preliminary study. 2nd SEMAT Workshop on a General Theory of Software Engineering (GTSE 2013) (2013) 51-58
31. Ambler, S.W., Lines, M.: Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. IBM Press (2012)
32. Wallace, L., Keil, M., Rai, A.: How Software Project Risk Affects Project Performance: An Investigation of the Dimensions of Risk and an Exploratory Model. Decision Sciences, Vol. 35 (2004) 289–321
33. Xu, P., Ramesh, B.: Using Process Tailoring to Manage Software Development Challenges. IT Professional, Vol. 10, No. 4 (2008) 39-45
34. Wiegers, K.: Software Requirements, 3rd Ed. Microsoft Press (2013)
35. Seacord, R.C., Plakosh, D., Lewis, G.A.: Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices. Addison Wesley (2003)